

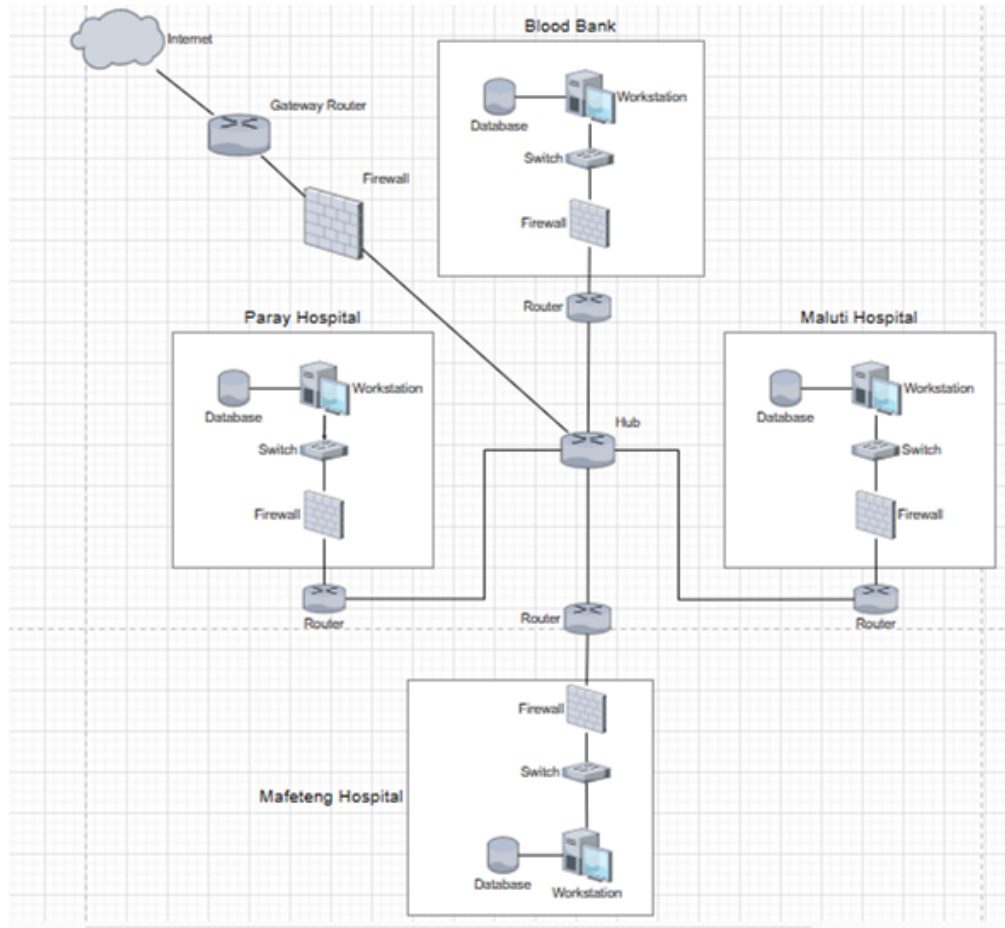


JINDAN

SYSTEM DESIGN

LIMPHO MAKULA 202000369
RETHABILE MAFA 201900216
KANANELO MOSHOSHOE 201800623
TS'ITSO MOTS'OEHLI 201902898
MPHO MAKALIANA 201700743

1. Detailed real world scenario system architecture diagram



Software architecture

The presentation layer is made up of three (3) components;

The load balancer; it receives requests from clients and distributes them across multiple web servers that host the user interface.

The web servers; they host the user interface of the application and serve requests from clients

The content delivery network (CDN); this is a network of servers around the world that cache static assets (images, videos and JavaScript files).

The application layer is made up of three (3) components;

Application servers; they are responsible for running the business logic of the application and handling requests from clients.

Message broker; it handles asynchronous communication between the application servers.
APIs; they are a set of protocols and tools for building software applications.

The database layer is made up of four (4) components;

Database servers: they are responsible for storing and managing the data in the system.

Database replication: this is the process of copying data from one database server to another.

Sharding: this is the process of partitioning data across multiple database servers.

Consistency and synchronization: these are important considerations in a distributed database system.

The software architecture would consist of a distributed database system that allows hospitals to access blood information from the blood bank and share blood amongst themselves. The platform would be based on a client-server model where the blood bank acts as the server and hospitals as clients. The application would be developed using a combination of programming languages, including Java and SQL, to provide a user-friendly interface and ensure efficient data management.

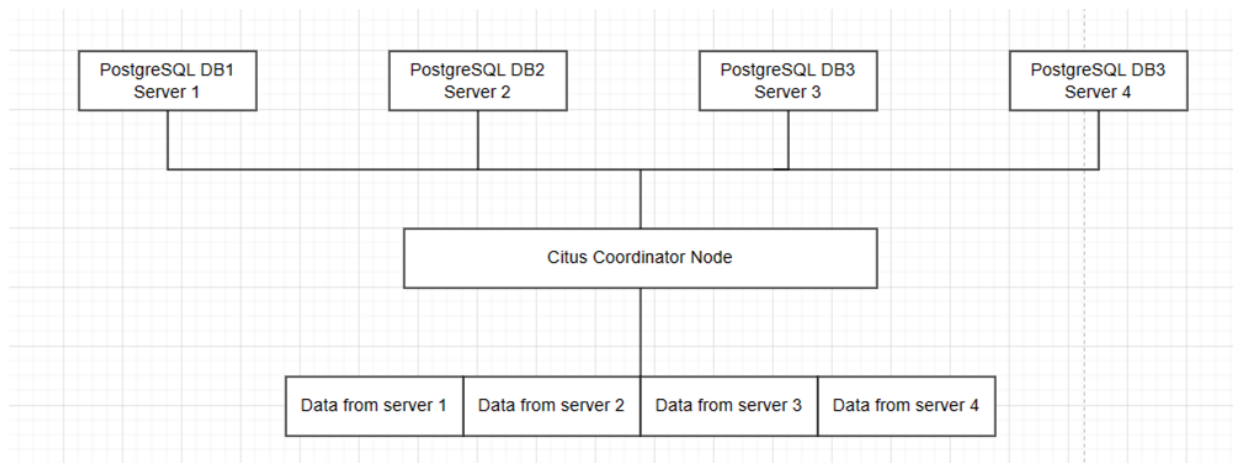
We'll also use Citus for implementation of distributed database management system.

Rationale:

We chose the hybrid system so that we can incorporate all possible architectures of our database. We took into consideration Lesotho's state; Lesotho has only one blood bank and multiple hospitals, this forces us to use a client-server architecture, as the blood bank acts as the server and hospitals as clients. Our system will also allow hospitals to act as servers (as they will supply other hospitals), meaning we will apply the multiple server-multiple clients architecture.

We also chose the star topology because it provides a centralized architecture where central location can manage and control data traffic between remote locations, even the traffic that goes in to different locations can be filtered according to the needs of each specific local site. The firewalls are there to protect the local sites from external threats and also from the external threat from the internet.

Detailed demonstration system architecture diagram

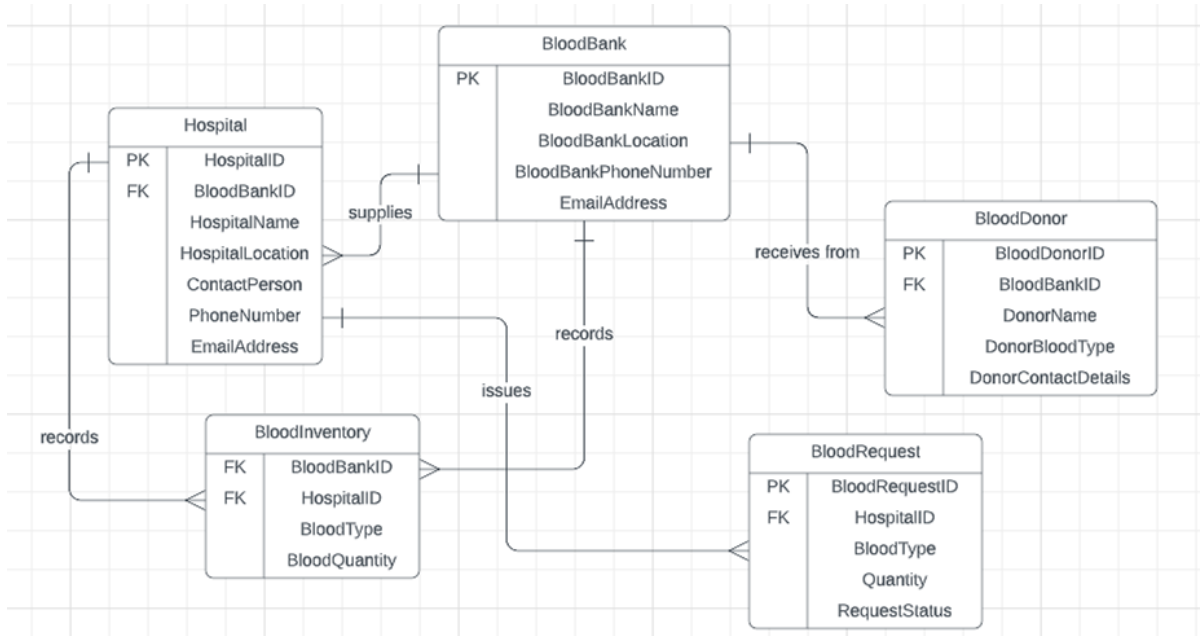


This diagram illustrates how we can use Citus to horizontally scale a database by distributing data across multiple nodes, and how citus manages the coordination and routing of queries to improve performance and scalability for large scale applications.

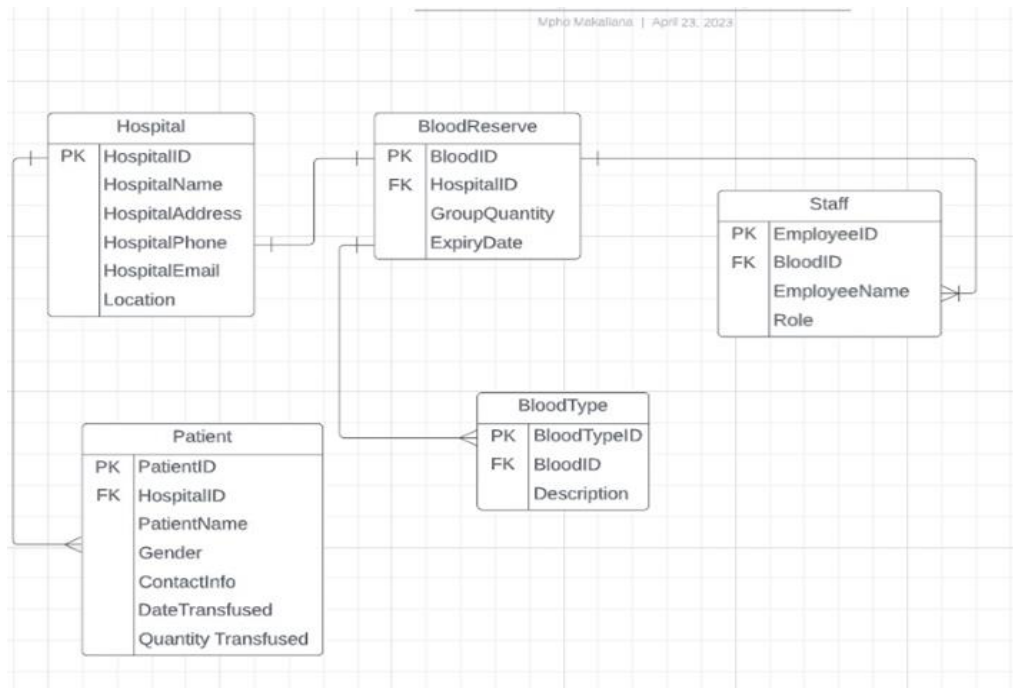
The citus coordinator node will acts as the coordinator for the cluster, managing metadata and routing queries to the appropriate nodes. PostgreSQL databases labeled Server 1 to server 4 each contains different sets of data or different parts of the same dataset. The distributed table will allow us to horizontally partition data and distributes it across multiple nodes, using the specific shard key that will be specified for each database.

3. Detailed distributed database design

Entity Relation Diagram of Distributed System



Entity Relation Diagram for each Hospital's Database



Assume there are two queries

1. The first finds the blood inventory with quantity:

```
SELECT * FROM BloodInventory
WHERE Quantity > 50
```

2. SELECT *
FROM BloodInventory
WHERE BloodType = 'value'
Blood types categorized into four main groups namely:
1. A
 2. B
 3. AB
 4. O

Simple predicates:

From application (1) (Q = Quantity)

P1 : Quantity > 50

P2 : Quantity <= 50

From application (2) (BT = BloodType)

P3 : BloodType = 'A'

P4 : BloodType = 'B'

P5 : BloodType = 'AB'

P6 : BloodType = 'O'

Using COM_MIN, we generate a set of complete & minimal simple predicates:

Pr' = Pr = {p1, p2, p3, p4, p5, p6}
= {Q <= 50, Q > 50, BT = 'A', BT = 'B', BT = 'AB', BT = 'O'}

We then generate the minterm predicates that form M:

M1 : (Quantity > 50) AND (BloodType = 'A')

M2 : (Quantity <= 50) AND (BloodType = 'A')

M3 : (Quantity > 50) AND (BloodType = 'B')

M4 : (Quantity <= 50) AND (BloodType = 'B')

M5 : (Quantity > 50) AND (BloodType = 'AB')

M6 : (Quantity <= 50) AND (BloodType = 'AB')

M7 : (Quantity > 50) AND (BloodType = 'O')

M8 : (Quantity <= 50) AND (BloodType = 'O')

Finally, the relation 'BloodInventory' is fragmented horizontally using M1 into the following fragments:

$$F_{BloodInventory} = \{BT_1, BT_2, BT_3, BT_4, BT_5, BT_6, BT_7, BT_8\}$$

- Data allocation including replication scheme
- Summarize in detailed diagrams and include few sentences to provide a rationale for your design choices

Data Allocation and replication scheme

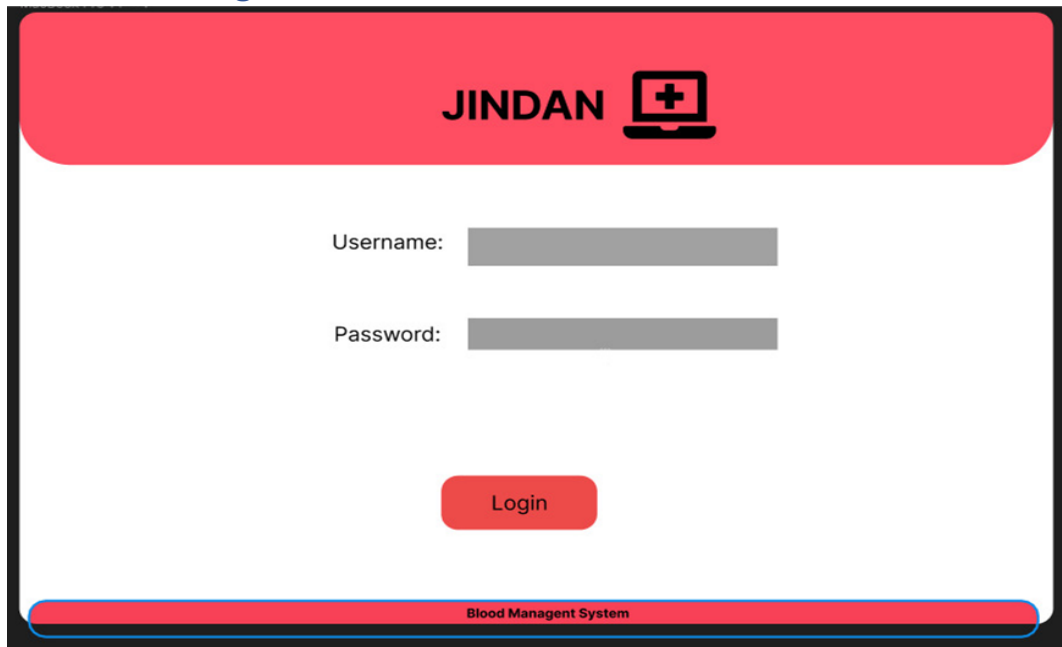
In-order to enhance availability and other factors which include fault tolerance, the following data will be replicated:

- blood group
- Quantity of the blood
- expiry date of the blood
- location (where the blood is located at moment of query issuing)


For confidentiality purposes, information about the donors will not be replicated; it will only be available at the blood bank. Since hospitals must be informed on the availability of blood at all times, data that will be replicated in the hospitals' databases will be the available blood groups and their quantities.

We will use the 'master-slave replication' scheme, whereby the blood bank will be responsible for all the updates to data, which will then be propagated to hospital nodes that maintain the copy of the data.

4. UI/UX Design



The image shows a login page for a system named JINDAN. At the top, there is a red header bar with the text "JINDAN" and a logo of a laptop with a plus sign. Below the header, the page has a white background. There are two input fields: "Username:" and "Password:", each followed by a gray rectangular input box. Below these fields is a red button with the text "Login". At the bottom of the page, there is a red footer bar with the text "Blood Managent System".

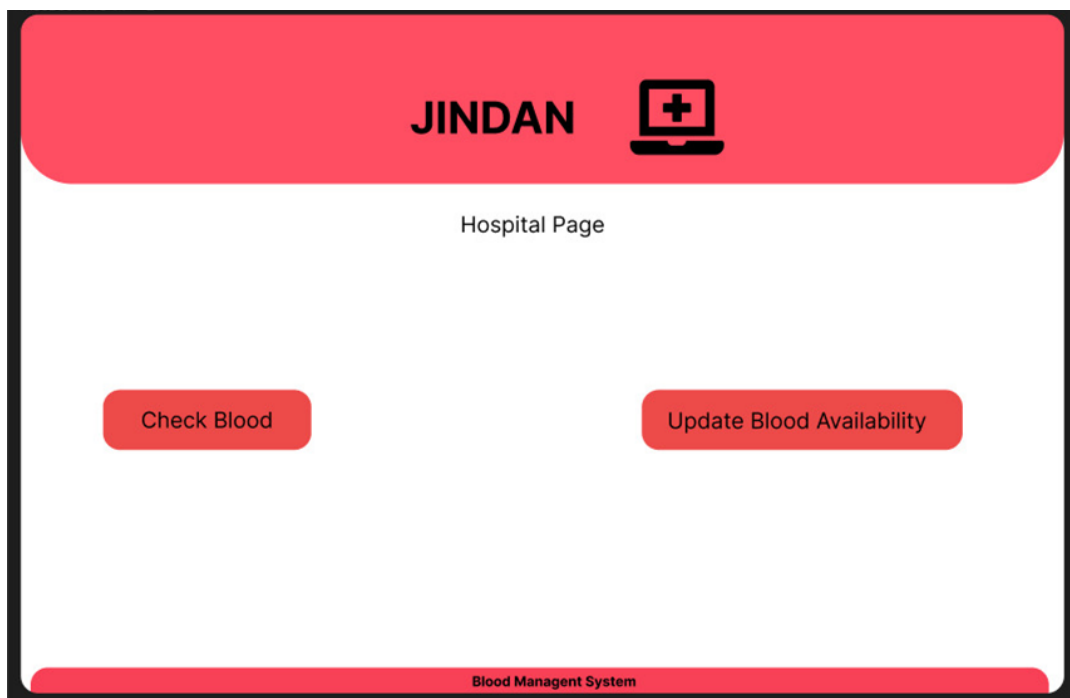
JINDAN 

Username:


Password:

Login

Blood Managent System



The image shows a hospital page for the JINDAN system. It features a red header bar with "JINDAN" and the laptop logo. Below the header, the page has a white background. The text "Hospital Page" is centered. There are two red buttons: "Check Blood" and "Update Blood Availability". At the bottom, there is a red footer bar with the text "Blood Managent System".

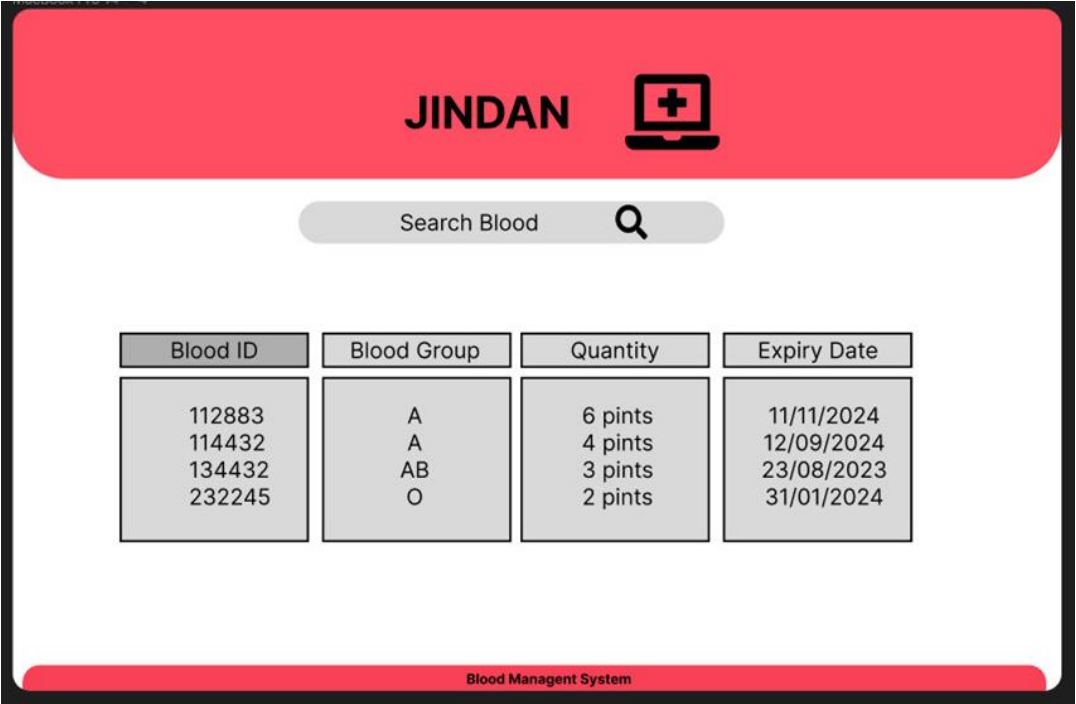
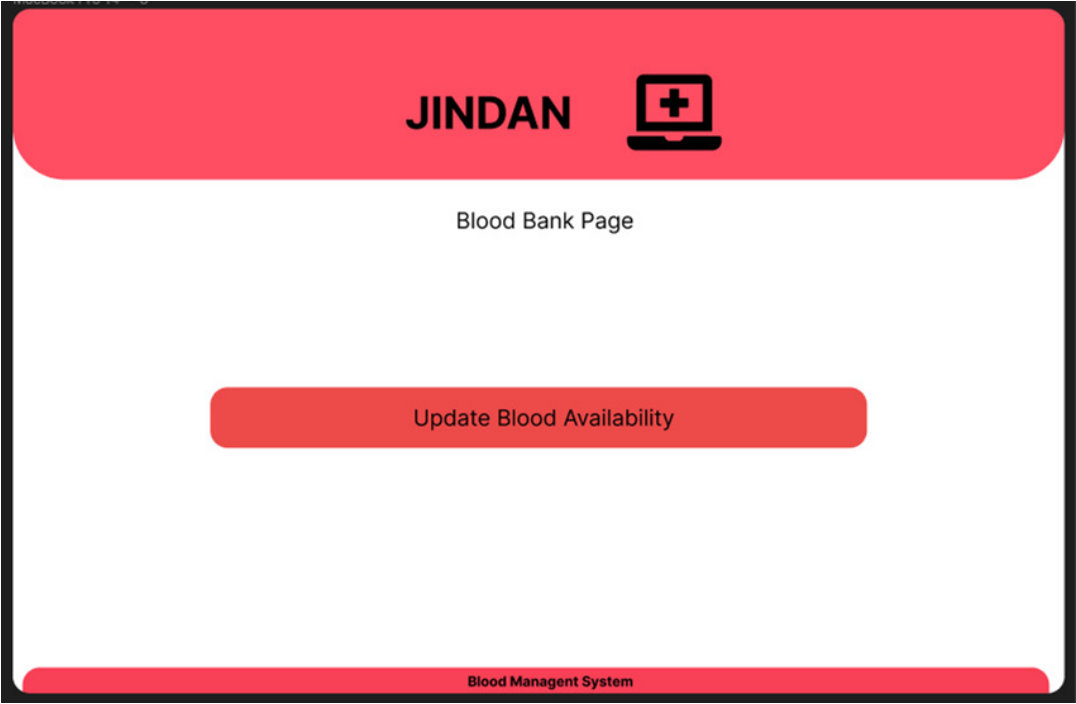
JINDAN 

Hospital Page

Check Blood

Update Blood Availability

Blood Managent System



JINDAN



Blood ID	Blood Group	Quantity	Expiry Date
112883	A	6 pints	11/11/2024
114432	A	4 pints	12/09/2024
134432	AB	3 pints	23/08/2023
232245	O	2 pints	31/01/2024

Save Changes

Blood Managent System

JINDAN



Search by Blood Group:



Blood ID	Blood Group	Quantity	Expiry Date	Location
112883	A	6 pints	11/11/2024	Paray Hospital
114432	A	4 pints	12/09/2024	Maluti Adventist
134432	AB	3 pints	23/08/2023	Mafeteng Hospital
232245	O	2 pints	31/01/2024	Ts'epong Hospital

Make Request

Blood Managent System

5. Detailed test plan

Facility	Tests to Run
Blood Bank	<ul style="list-style-type: none">-Check the quantity of blood at the different hospitals.-Update blood quantity.
Hospitals	<ul style="list-style-type: none">-Update the availability of blood.-Check the available blood quantity.-Check blood and other hospitals.